Game strategies for The Settlers of Catan

Game strategies for *The Settlers of Catan*

Markus Guhe, Alex Lascarides
School of Informatics
University of Edinburgh
Edinburgh EH8 9AB
Scotland
Email: m.guhe@ed.ac.uk, alex@inf.ed.ac.uk

***Abstract*—We present an empirical framework for testing game strategies in *The Settlers of Catan*, a complex win–lose game that lacks any analytic solution. This framework provides the means to change different components of an autonomous agent's strategy, and to test them in suitably controlled ways via performance metrics in game simulations and via comparisons of the agent's behaviours with those exhibited in a corpus of humans playing the game. We provide changes to the game strategy that not only improve the agent's strength, but corpus analysis shows that they also bring the agent closer to a model of human players.**

I. Introduction

Rational action, according to Savage, is typically defined as that which maximises the agent's *expected utilities*—an optimal trade-off between what he *prefers* (typically defined by a utility function) and what he *believes* he can achieve (typically defined via a dynamic Bayesian network; [13]). Solving a game problem involves finding *equilibrium strategies*: an optimal action for each player in that it maximises his expected utility, assuming that the other players perform their specified action [14]. But this Savagean model is highly idealised and humans often deviate from it [2], [9], especially in complex games. *The Settlers of Catan* (or *Settlers*, [17]; `http://catan.com`) is one such game. Even if *Settlers* had an analytic solution, it wouldn't necessarily match what humans do. Further, its game tree isn't surveyable—e.g., the options for negotiating trades in natural language aren't bounded [3]—and so algorithms like backwards induction for computing expected utilities don't work on their own.

One response to this is to develop a *symbolic* model consisting of *heuristic* strategies for playing the game. Developing such models potentially has two advantages. First, a symbolic model can in principle lead to an interpretable model of human expert play, but this requires a means for comparing and evaluating the performance of the model with that of human experts, and using evaluation to guide further developments and improvements. Second, a symbolic model can provide a *prior distribution* over which next move is likely to be optimal, and this is critical to the success of using current machine learning techniques on complex games—where by success we mean that the posterior distribution over optimal actions acquired through training improves on the baseline prior distribution. For instance, both reinforcement learning (RL; [15]) and the online learning technique Monte Carlo tree search (MCTS; [4]) have been used to train agents to play *Settlers*. But in both cases, they relied on the agent having *prior* to its learning phase a decent game strategy: Pfeiffer [10], [11] and Hanna and collaborators [8] demonstrate that RL succeeds only when they use substantial prior symbolic heuristics and a suitable, domain-specific internal structure for restricting search; and Roelofs [12] and Szita *et al.* [16] show that deploying MCTS likewise needs a prior distribution that already captures sophisticated and domain-specific strategies for restricting exploration.

The purpose of this paper is to provide an empirical framework for developing and improving symbolic strategies for playing complex games that realises the above two advantages of symbolic modelling. Like the work we just mentioned our domain is *Settlers*, and our starting point is an existing open source symbolic agent `JSettlers` [18]. We show here that by using performance data gathered from game simulations among agents and quantitative comparisons of that data with data that is extracted from a corpus of humans playing the game [1], one can guide the design, implementation and evaluation of modifications to the agent, and so achieve relatively large improvements in the agent's performance on the game—the agent we have developed using this methodology wins 43% of the games when playing 3 of the `JSettlers` agents that we started with (called *jsettlers* agents below). Furthermore, the behaviour of our modified agent, though still different from the human behaviour exhibited in the corpus, is demonstrably closer to it than the `JSettlers` agent is.

As part of a project on strategic negotiation as it occurs in complex games, we are investigating *Settlers* as an example domain [5], [7]. In [7] we focussed on how human errors in beliefs—in particular forgetting—impact negotiating and trading in *Settlers*, and we took the first steps towards designing negotiation strategies that compensate for deficiencies in beliefs, and in [6] we address the benefits of persuasion during negotiation in *Settlers*. In this paper we report on improvements to other parts of the game strategy: in particular, what (and where) to build on the board.

There are several empirical approaches to modelling *Settlers*, but none of them deal with the full game since they ignore the options of negotiating and trading with opponents. As we mentioned earlier, the MCTS approach used in [12] and [16] and the RL approach of Pfeiffer's [10], [11] all show that training must build upon what is already a complex prior strategy that incorporates sophisticated and domain-specific reasoning. Hanna *et al.* [8] get the same result for the *Fungus Eater* game: the models learning best and quickest use the most hand-coded knowledge and most refined architecture for subdividing the task in a domain-specific way. The *Fungus Eater* game has a large state space, but relatively few actions

compared to *Settlers*, and it is a single agent environment so opponents' preferences don't count. None of these approaches evaluate whether their prior models are cognitively plausible, and the symbolic models are built on an ad hoc basis. Here, we prove by demonstration that it is possible to build a game simulation environment that supports the systematic and controlled evaluation of symbolic models, which in turn justifies and guides updates to them that result in improved player performance, and our empirical method also provides the means to compare the agents' behaviours within game simulations against those of people. We hope that in future work we can show that improving the prior leads to improved results from training (both with RL and with MCTS).

II. The Settlers of Catan

The Settlers of Catan is a complex multiplayer board game; the board is a map of *Catan* consisting of hexes of different types: hills, mountains, meadows, fields and forests (cf. Fig. 1, [17], and `http://catan.com`). Two to four players settle on *Catan* by building settlements and cities connected by roads. It is a zero-sum game: the first player to get 10 Victory Points wins, all others lose. Players obtain Victory Points by:

- Building a settlement (1 point),
- Upgrading a settlement to a city (1 additional point),
- Drawing a Victory Point development card (1 point),
- Building the Longest Road (at least 5 consecutive roads, 2 points),
- Assembling the Largest Army (by playing at least 3 Knight development cards, 2 points).

Performing these actions needs resources—clay, wood, rock, ore and sheep—in different combinations. A different number of resources is needed for different pieces (e.g., 2 for a road, 5 for a city), but some types of resources are more frequent than others: for instance, there are 4 hexes for sheep but only 3 for ore, and building a city requires one to build a settlement first. So calculating the most likely time it will take to build a given piece on the board given one's beliefs about the current game state is very complex, and human calculations will be error-prone.

The game starts with a setup phase where each player in turn places two settlements and two roads. Players then take turns and attempt to obtain resources and to build pieces. A turn starts with the roll of dice. So, future game states are non-deterministic and players must calculate the risks of moves. Each player potentially obtains or loses resources through dice rolls. A player with a settlement or city touching a hex with the number rolled, cf. Fig. 1, receives the corresponding resource (1 for a settlement, 2 for a city): hills produce clay, mountains ore, meadows sheep, fields wheat and forests wood. The desert hex produces no resource.

Losing resources occurs when the dice roll is 7; this activates the robber. (Because 2 six-sided dice are used, results of dice rolls are not distributed evenly, and 7 is the most frequent result.) The player whose turn it is moves the robber to a new hex and 'robs' one of the players with a settlement or city adjacent to this hex—he gets one (random) resource

Fig. 1. A game of *Settlers* in `JSettlers`.

from the player he chooses to rob. This hex then produces no resources until the robber is moved elsewhere. Any player with more than 7 resources must discard half of them. The robber can also be moved by playing a Knight card (a type of development card). The robber starts the game on the desert hex. What gets robbed is hidden to the unaffected players. Thus *Settlers* yields partially observable game states: not only are the preferences of opponents hidden, but so are their resources.

The player whose turn it is can additionally trade resources with the bank at a rate of 4:1, ports at a rate of 3:1 or 2:1 (depending on the port), or with other players (in any way they agree to, except for simply giving resources away).

In addition to the 5 Victory Point development cards and the 14 Knight cards, there are 6 Progress development cards:

- Road Building: immediately build 2 roads
- Year of Plenty: get 2 freely chosen resources from the bank
- Monopoly: get all of the chosen resources of one type from the other players

III. Planning in JSettlers

We use an open source implementation called `JSettlers` (`http://jsettlers2.sourceforge.net`, [18]; we branched off version 1.1.12). `JSettlers` is a client–server system: a server maintains the game state and passes messages between each of the players' clients. Clients can be human players or computer agents.

The core idea of the game strategy used by the `JSettlers` computer agent is conceptualising the game as a race between the players to be the first to gain 10 Victory Points [18]. Consequently, the valuation functions that determine the relative merit of a particular game move calculate

how much the move reduces the estimated time needed to get 10 Victory Points.

The `JSettlers` agent comes with two inbuilt game strategies: a *fast* strategy and a *smart* strategy. The fast strategy solely relies on minimising the *Estimated Time to Build* (ETB), i.e. on finding the game action that enables the player to build the next piece in order to get the next Victory Point. In contrast, the smart game strategy evaluates actions beyond obtaining the next Victory Point. Below, we will provide results via game simulations which measure the relative merits of both approaches.

The `JSettlers` agent goes through multiple phases after the dice roll that starts his turn:

1) Deal with game events: e.g. placing the robber; acquiring or discarding resources.
2) Determine legal and potential places to build pieces.
3) Compute the *Estimated Time to Build* pieces on legal places (ETB) and the *Estimated Speedup* (ES), i.e. by how much building a piece, e.g. a settlement, in a particular location reduces the *Estimated Time to Win* (ETW) the game.
4) Compute the *Best Build Plan* (BBP), a (sequence of) build actions that best advances the agent's position in the game.
5) Try to execute the BBP, including negotiating and trading with other players and/or the bank or a port.

A detailed description of all steps is given in [18]; for our purposes here it suffices to point out that the main factor in choosing the BBP is the ETB, and the fast strategy considers only the ETB of the *next* piece, with ES (Estimated Speedup) used as a tiebreaker (so it aims to minimise ETB and maximise ES). These values are calculated on the average number of resources a player can expect to receive from the hexes through dice rolls: e.g. if a player wants to build a road (costing one wood and one clay) and he already has wood, then the ETB is the estimated time to get clay via dice rolls or trades with the bank or a port.

The `JSettlers` agent generates 5 different types of build plans: Settlement (including roads needed), City, Development Card, Longest Road and Largest Army. So, there is no Road build plan as such, and Development Card and Largest Army both map to the action *buy a development card*.

In [7] and [6] we focus on negotiating strategies; i.e., a subset of step 5. In this paper, we investigate and improve the strategies in steps 3 (estimates of the relative preferences over build plans) and 4 (identifying an optimal sequence of build actions for achieving a win). But our changes also substantially improve step 5, and critical to these successes is our ability to perform game simulations where we can control arbitrary aspects of the agent's policies so as to perform sound evaluations of specific hypotheses about optimal play.

IV. Measuring agent performance

We employ multiple measures to assess an agent's performance. First, the **win rate** reflects an agent's overall success: after all, planning, building, negotiating and trading are just means to win the game. A simulation consists of 10,000 games, each with 4 agents, where we pit 1 modified agent against 3 baseline agents (benchmark or ranking, see below). Therefore, the null-hypothesis is that each agent wins 25% of the games. We use 10,000 games to avoid the arbitrary behaviour engendered by small numbers, given the degree of non-determinism in the game. This number of games per simulation roughly means that with a significance threshold of $p < 0.01$, win rates between 0.24 and 0.26 are not significantly different from the null-hypothesis. Each simulation takes about 1 hour on a current desktop computer.

The other measures fall into two categories: those concerning the **acquisition of resources**; and those concerning **building**. The null-hypotheses here are that all agents produce the same values. We measure the quality of an agent's strategy for acquiring resources in several ways. First, we consider how many resources he obtains through *trading* and how many via the *hexes*. Obtaining a resource from a hex comes from smart placement of pieces on the board (and so is also a measure of the quality of one's build plans). Smart placement also contributes to obtaining resources via trading (via ports, and because trades with banks require 4 resources). Obtaining a resource via trading also stems from successful *negotiating* with other players, and the more resources a player has, the better his chances to negotiate suitable trades. Note that trading is less desirable, because the player has to give away resources in exchange.

We measure the quality of different **build strategies** by the number of:

- Build plans (grouped by the 5 different types),
- Pieces actually built (road, settlement, city, development card), excluding the 2 roads and 2 settlements that are placed in the setup phase of the game,
- Resources received from hexes (see above).

The proportion of planned builds to actual builds is a rough measure of how accurately agents assess what they can achieve. Measuring the number of resources received from hexes (by dice rolls) is a measure of the quality of the build strategy for the stated reasons. Note this number does not include resources gained from playing development cards (Knight, Year of Plenty, Monopoly).

We use *Z*-tests for testing the win rate and pairwise *t*-tests for the number of resources received. We will mark significant differences to the null-hypotheses for the modified agents in bold. We do not report differences between the 3 baseline agents that the modified agent plays against: there were no significant differences between them. We will also give results for the pieces the agents built, but because these are distributions with a very small N, there is no useful statistical significance test. We did, however, compute the Kullback-Leibler divergence statistics for these data (all were greater than 0.5), but because there is no generally agreed on interpretation of how large a distance constitutes a sizeable difference, it is open to interpretation on how different agents are with respect to these measures. *t*-tests on the absolute numbers show they are significantly different, but because winning agents build more pieces on average, this is not a surprising outcome.

Correlation between win rate and resources from hexes.

Intuitively, there should be a close correlation between the number of resources an agent receives from hexes (i.e. via dice rolls) and its win rate; more precisely: the more resources an agent gets from hexes, the higher his win rate should be. Our simulations are supporting evidence for this intuition: Figure 2 depicts the correlation between the win rate and the number of resources the agent gained on average from the hexes on the board. This correlation is significant ($r = 0.207, p < 0.05, n = 99$). Thus, the more resources an agent gets from hexes, the better his chances to win.

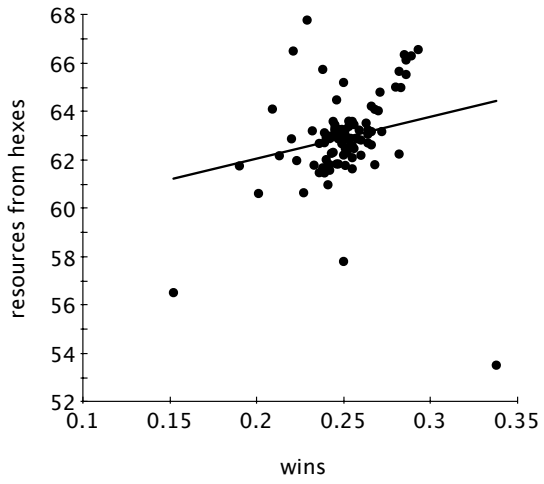

Fig. 2. Dependency between wins and resources gained from hexes.

V. Build strategies for *The Settlers of Catan*

We looked at the most interesting parts of the agents' build strategies:

- **Initial placement**, i.e. placing settlements and roads during the setup phase,
- Choosing among the different types of **build plans** at any given point in the game after the setup phase, and
- The potential disadvantage of the limited **lookahead** of the myopic 'fast' strategy to the elaborate `JSettlers` 'smart' strategy.

Since we make no changes here to the trading strategy, all differences in resources received through trading are effects of changes to the build plans that agents choose.

A. Improvements to initial placement

An agent that uses a more principled approach for placing settlements and roads in the setup phase (the agent we are calling *benchmark* because apart from changes to initial placements it uses the original `JSettlers` algorithms and is one of our baseline agents) is a substantial improvement over the agent that comes with the `JSettlers` system (the agent we are therefore calling *jsettlers*), even though their strategies are identical after the setup phase.

While the jsettlers agent's decisions about initial placement are dependent only on which locations on the board generate the most resources (so that the sum of the estimated time to build each of the 4 types of pieces – road, settlement, city, development card – is minimised [18, pp. 87–93]), our new algorithm uses a comprehensive weighting function that:

1) Disfavours hexes that have the same number (because occupying hexes with different numbers enables the player to receive resources with a more even spread, which increases the likelihood that resources are available in every turn and decreases the likelihood of having to discard resources when a 7 is rolled),
2) Disfavours building both settlements on the same hex (which is a special case of the first factor, but this is disfavoured more strongly),
3) Uses a weighted scoring function that uses different weights for the resource combinations needed to build the 4 different types of pieces (effectively, the fewer resources are needed for a piece, the more it is favoured; this contrasts with the jsettlers agent, who is indifferent about which of the 4 types of pieces he will be able to build fastest while considering initial placement).

One jsettlers agent playing against 3 of our 'benchmark' baseline agents only wins 15.2% of the games. This demonstrates that decisions about the initial setup are critical, because it opens up or limits the player's subsequent options. (In all result tables we give the modified agent first and the averages for the 3 baseline agents second.)

		resources		
agent	wins	trade	hex	sum
jsettlers	**0.152**	**6.0**	**56.5**	**62.5**
benchmark	0.283	7.6	65.0	72.6

The benchmark agent also receives about 15% more resources from the hexes (by dice rolls) than the jsettlers agent. This means that it has not only many more opportunities to build pieces directly but also that it has more opportunities and options to trade. The table below gives the average number of trade offers the agents make per game, their successful offers and total number of trades (which include accepting trade offers by other agents.)

	trades		
agent	offers	succ. offers	total trades
jsettlers	**6.2**	**1.5**	**5.2**
benchmark	13.8	3.8	6.8

Because after the setup phase the agents use the same build strategy, they do not differ with respect to the pieces they build.

B. Ranking build plans

A particular weakness of the planning algorithm that computes the BBP used in the jsettlers and benchmark agents is that they use a nested sequence of heuristics that, by design, incorporates a global default preference for certain types of build plans over others: *City* $\succ$ *Settlement* $\succ$ *Largest Army* $\succ$ *Longest Road* $\succ$ *Development Card*. The algorithm selects the highest build plan in this list that has an ETB lower than the next build plan in the list. Additionally, the agent sometimes (though not always) does not search the whole list and skips

over build plans lower in the list, even though their ETB may be lowest and indeed may contribute more, relatively speaking, towards an overall plan to win. This is particularly true for development cards.

We replaced this global default preference ordering with a simple ranking function that decides among the plans solely by ETB (and ES as a tiebreaker for plans with the same ETB). In other words, preferences over *all* options are conditioned on some aspects of the current game state. The conjecture is that this makes decisions about optimal build plans more context sensitive, removing certain default assumptions about the desirability of certain build plan types. Our *ranking* agent adopts this ranking function to approximate the expected utility of a build plan; the functions for computing ETB and ES were left unchanged.

Apart from the win rate, the main difference between the ranking agent and its benchmark opponents is the types of build plans they choose. While the benchmark agent hardly ever chooses to buy a development card (thanks to its default global dispreference), the ranking agent uses an almost even split between this and building settlements and cities. (Note that these numbers show the *proportion* of build plan types, not the absolute counts.)

		build plans (proportion)				
agent	wins	settlm.	city	card	LA	LR
ranking	**0.299**	0.35	0.30	0.29	0.04	0.02
benchmark	0.234	0.43	0.44	0.00	0.08	0.05

This distribution in the chosen plans is reflected in the types of pieces that the agents actually build: the ranking agent buys development cards in 35% of the cases.

		pieces built (proportion)			
agent	wins	roads	settlm.	city	card
ranking	**0.299**	0.27	0.18	0.20	0.35
benchmark	0.234	0.44	0.24	0.28	0.05

As already mentioned, there is no statistical method to test whether these differences are significant, but the KL divergence for the distributions of build plans for the ranking agent vs. its benchmark opponents is 6.22 and for the pieces built it is 0.63.

C. (Dis-)favouring types of build plans

Because buying development cards is the main difference between the benchmark and the ranking agent and because this gave the ranking agent a substantial advantage, we explored whether we can further improve on the build strategy by favouring or disfavouring one of the possible 5 build plan types, i.e. reducing or increasing the number of times the agent chooses a particular type of build plan.

We therefore manipulated the ETB with a *favouring factor* (fav) according to the following formula:

$$etb_{scaled} = etb_{orig} - (fav * etb_{orig})$$

with etb_{orig} as the original estimated time to build and etb_{scaled} the estimated time to build scaled with the favouring factor fav. Note that:

- Positive values for fav favour the corresponding build plan type, because they reduce the ETB.
- $fav \leq 1$, because the ETB cannot be a negative value.
- If $etb_{orig} = 0$, then it etb_{scaled} remains 0, so ETBs for build plans that the agent can execute immediately are not affected.

1) Development Cards.: Using this formula to favour buying development cards indeed changes the agent's performance. Fig. 3 shows how win rate, Development Card chosen as BBP per game and development cards actually bought per game change with the favouring factor when playing this agent against 3 unmodified ranking agents. The grey area indicates when win rates are not significantly different from the null-hypothesis (i.e., to win 25% of the games). Since the win rate rises to 0.293 by *dis*favouring development cards, a strategy solely relying on the ranking agent's ETB now *over*estimates their usefulness.

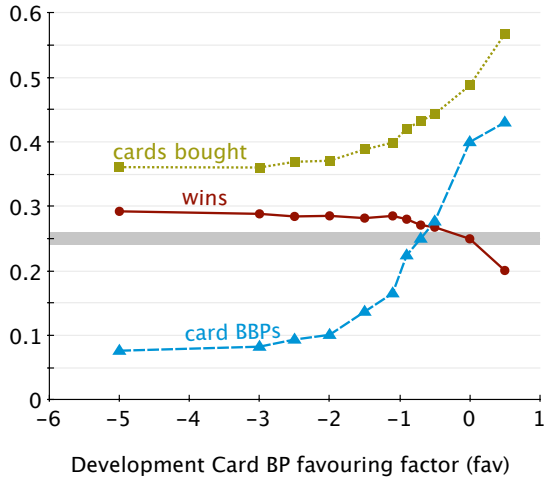

Fig. 3. Effects of favouring Development Cards build plans. The grey area indicates win rates that do not differ significantly from the null-hypothesis.

While the total number of resources gained per game does not change much across the different agents, ranking agents that disfavour development cards have an advantage because they receive more resources from hexes directly, for which they don't have to give away resources in exchange (whereas for traded resources they do).

		resources		
fav	wins	trade	hex	sum
–5.0	**0.293**	**15.11**	**66.57**	**81.68**
(ranking)	0.236	16.81	61.54	78.35
–0.5	**0.268**	**16.45**	**64.10**	**80.55**
(ranking)	0.244	17.57	62.32	79.89
0.0 (ranking)	0.250	16.55	63.17	79.72
0.5	**0.201**	**19.54**	**60.62**	**80.16**
(ranking)	0.266	19.04	64.23	83.27

So, the agent using $fav_{cards} = -5$ is our best-performing agent overall. If we run this agent against the jsettlers agent that originally comes with the `JSettlers` system, he achieves a win rate of 0.430. We will be calling this agent *favDC* in the following.

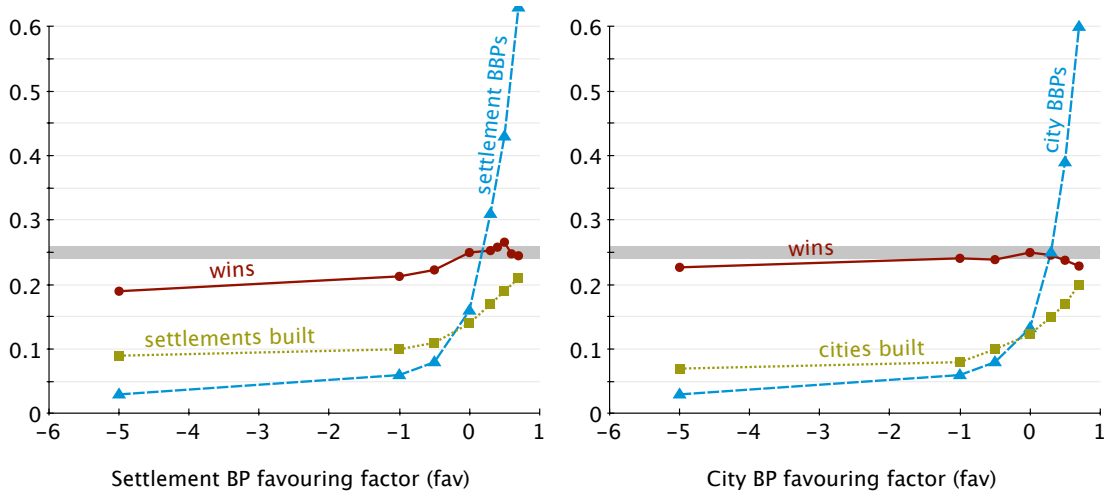

Fig. 4. Effects of favouring Settlement or City build plans. The grey area indicates win rates that do not differ significantly from the null-hypothesis.

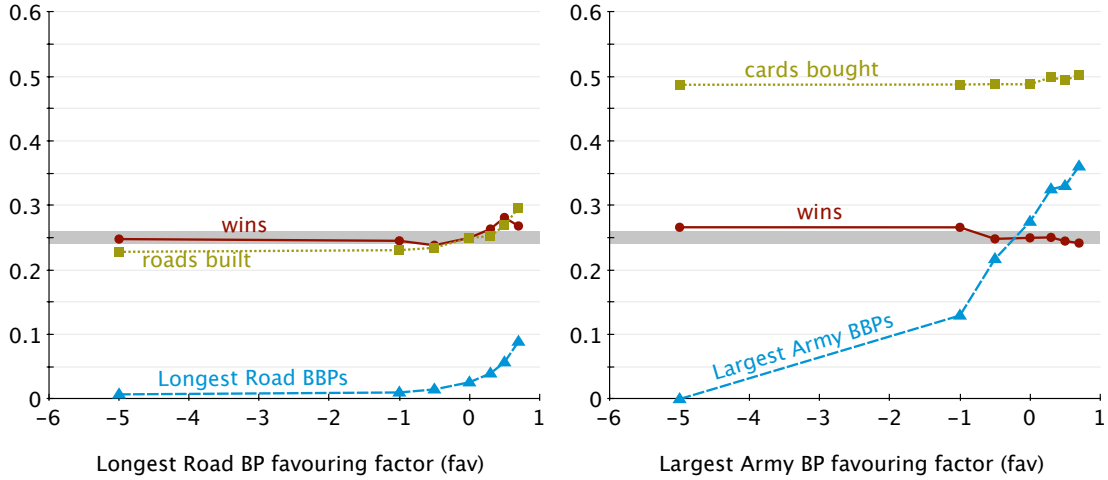

Fig. 5. Effects of favouring Largest Army and Longest Road build plans. The grey area indicates win rates that do not differ significantly from the null-hypothesis.

2) Settlements and Cities.: Applying the favouring factor to settlements and cities, cf. Fig. 4, shows that they cannot be improved to the same extent. The only (slight) improvement in win rate occurs when favouring Settlement build plans with $fav = 0.5$ (win rate is 0.266). This suggests that the agents' estimation of the utilities for these build plans is more accurate than that for development cards.

3) Longest Road and Largest Army.: Positive favouring factors for Longest Road build plans and negative ones for Largest Army build plans lead to some improvements, cf. Fig 5. The biggest improvement here is favouring Longest Road with a factor of 0.5, which increases the win rate to 0.282. While this agent does not actually build many more roads overall (0.269 vs. 0.250), he chooses Longest Road build plans about twice as often as the ranking agent (0.026 vs. 0.057).

The biggest improvement for favouring Largest Army build plans occurs with the favouring factor −1 (win rate: 0.266). Because the build action for the Largest Army build plan is identical to the Development Card build plan (buying a development card) and because the change in win rate for disfavouring development cards is very large, the change is most likely due to the different number of attempts the agent makes to buy development cards.

D. Fast vs. smart

Unlike the fast agent who just plans how quickly it can build the next piece (by minimising the ETB), the smart agent plans ahead until the end of the game by creating the best sequence of build plans that will get it to 10 Victory Points in the shortest *Estimated Time to Win* (ETW). Computationally, this is very costly (the smart agent is slower by a factor of approximately 30). So, we tested whether an agent actually profits from this additional cost.

When running 1 *smart benchmark* agent—i.e. an agent that uses improved initial placement and the smart game strategy—against 3 (fast) benchmark agents, the smart agent has a win rate of 0.338. Looking at the types of pieces they build, the main difference between the agents is, again, that the fast agent is buying almost no development cards.

agent	wins	pieces built (proportion) roads	settlm.	city	card
smart benchmark	**0.338**	0.33	0.14	0.14	0.39
(fast) benchmark	0.221	0.46	0.25	0.28	0.01

An unexpected result in this simulation is that while the fast benchmark agent receives 66.5 resources from hexes per game, the smart agent receives only 53.5 per game. The smart agent's higher win rate suggests he uses his resources more effectively than a fast agent does.

So far, we have seen that both the smart and the ranking agents buy more development cards than the benchmark agent. Further, playing 1 smart benchmark agent against 3 ranking agents has the smart agent *lose* quite decisively; his win rate is only 0.196.

agent	wins	pieces built (proportion) roads	settlm.	city	card
smart benchmark	**0.196**	0.36	0.16	0.15	0.32
ranking	0.268	0.22	0.13	0.12	0.53

Furthermore, the smart benchmark agent, again, gets fewer resources overall, but this time the difference is mainly due to the difference in number of resources received by trade:

agent	resources trade	hex	sum
smart benchmark	**12.2**	62.6	**74.8**
ranking	15.7	62.7	78.4

There are two further features of these simulations that are worth noting. Firstly, the ranking agent has only 1/30th of the smart agent's computational cost, and yet his win rate in the 1 vs. 3 configuration is significantly higher. If we use different configurations (3 smart vs. 1 ranking or 2 vs. 2), the win rates do not differ significantly from the null hypothesis, but again the smart agent does not profit from his time-consuming deliberations either.

The second feature is that in all the above simulations, the smart agent profits from our improved `JSettlers` protocol for negotiation, where the agent making a trade offer waits until he has received a response from all agents he made the offer to before deciding which agent to make the trade with (if multiple agents accept his offer). If we use the original `JSettlers` trading interaction, the first agent accepting a trade offer gets to make the trade, which in this context means it will always be one of the (fast) ranking agents. Here, the ranking agent has a win rate of 0.272 in our usual 1 (smart) vs. 3 (ranking) configuration, but now the ranking agent also has an advantage in the other configuration: in a 3 vs. 1 setting it is 0.273 for the ranking agent and even 0.289 a 2 vs. 2 configuration.

When we run our best-performing agent (favDC, the one disfavouring Development Card build plans with a factor of –5; again using our improved trade interaction), this effect becomes even stronger, and the agent substantially outperforms the smart agent (we only give the results for the 1 smart benchmark vs. 3 (fast) disfavouring development cards, and the 3 vs. 1 setup respectively in the table below).

agent	wins	pieces built (proportion) roads	stm.	city	card
smart benchmark	**0.191**	0.33	0.14	0.14	0.39
favDC	0.270	0.30	0.16	0.13	0.42
favDC	**0.323**	0.28	0.16	0.13	0.44
smart benchmark	0.226	0.32	0.14	0.15	0.39

agent	resources trade	hex	sum
smart benchmark	**11.4**	**57.9**	**69.3**
favDC	12.9	62.3	75.2
favDC	**11.8**	**64.0**	**75.8**
smart benchmark	14.1	59.6	73.7

So, summing up, the smart benchmark agent performs better than the fast benchmark agent, but the ranking agent performs better than both of them, and with the additional disfavouring of buying development cards, the ranking agent's performance becomes even better. So, our best performing agent is beating all other agents. This means that while there is some benefit to being smart (at the cost of speed), that benefit is outweighed by the benefits of the ranking and our best-performing agent *without* the cost of losing speed.

VI. Comparison with human data

In the following table, the row *corpus* shows the results from 38 games between human players collected as described in [1]. To obtain comparisons with the human corpus, we make an idealised assumption that the human players are of the same type, and so compare the human play against game simulations where *4 identical agents* play one another.

agent	pieces built (proportion, except *num*) roads	settlm.	city	card	num
corpus	*0.49*	*0.17*	*0.10*	*0.25*	*12.9*
favDC	0.30	0.18	0.15	0.36	12.3
ranking	0.25	0.14	0.12	0.49	12.5
smart benchmark	0.33	0.14	0.14	0.39	11.3
(fast) benchmark	0.46	0.24	0.28	0.01	10.2

According to the L1 distance metric, measured on both the number of pieces built and their proportions over the different types, our strongest agent (favDC, i.e., the one favouring development cards with a factor of –5) matches the human behaviour closest. Both this agent and the smart benchmark agent exhibit a similar KL divergence to human performance, with the KL measure for the smart benchmark being slightly smaller (though without a test to compare different KL measures we cannot know if this is significant). But KL divergence compares only the proportion of pieces built and ignores the absolute number of pieces built. Since the number of pieces is a critical aspect of one's behaviour—it reflects how efficiently an agent's pieces accrue points towards winning—the L1 metric is our preferred measure for comparing behaviours in this case, making favDC our best match. Thus, our manipulations generally improve the match between agents and corpus. This is proof by demonstration that by designing an agent whose strategies can be easily modified it is possible to rapidly converge to a model that closely matches observed human behaviour.

VII. Conclusions

We have demonstrated the feasibility of using a rigorous experimental setup, where performance data gained via game simulations and their comparison to data on human behaviour can be used to guide improvements to an already sophisticated symbolic model of a complex game. More precisely, our method consists of changing one aspect of an agent's strategy, running extensive simulations against other agents whose strategies control for other factors, and collecting comprehensive performance measures from these simulations so that these metrics can guide further development. We showed that in the domain of *The Settlers of Catan* at least, adapting the symbolic model to increase win rates correlates with getting closer to the performance profiles exhibited by human play.

We do not have an adequate model of a human player of *Settlers* yet. But we have identified the ingredients we need for this and produced an environment to pursue the issue by (1) running agent-only simulations, (2) observing people play and (3) playing mixed games between human players and agents as future work. This makes it possible to pursue our goal of creating a good model of human players, as well as building a very strong automated player, which (like for many complex games like the AI favourites chess or Go) offers us many new insights into intelligent behaviour.

With our work we have also shown that there is a danger when deploying a decision tree approach to decision making that fixes (default) preferences in a way that abstracts too much away from the specific context of the move. This is a natural risk with these approaches because their strength (to overcome the complexity in reasoning by making default assumptions) is also their weakness (default assumptions run the risk of ignoring properties of the current state that are critical to an effective decision). Through a comparison of our original model with human behaviour, we found that a more flexible method was needed for evaluating the relative preferences of each action: specifically, a comparison with human data exposed the fact that the benchmark agent was deliberating about optimal action in a way that under-estimates the benefit for development cards and over-estimates that of other pieces. Thus we achieved one major improvement to the agent's performance by relaxing its default preference and instead comparing all options via a reward function that draws on quantitative values via features of the current game state (in effect, this is what ETB and ES do).

In future work, we plan to explore other aspects of the game strategy, e.g., negotiating and robbing, and we will address one limitation of the agent's current (fast) approach to determine BBPs, viz. that it doesn't adequately balance a higher ES against a longer ETB—simulations show that frequently switching the high-level strategy is problematic (cf. also [8]). Once these enhancements are complete, we will release the various agents and game simulation environment in an open source repository. We also plan to investigate whether an improved prior symbolic model enhances the learning process in systems that learn optimal play via both RL and MCTS.

Acknowledgment

This work is supported by ERC grant 269427 (STAC). We would like to thank Kevin O'Connor for implementing the setup protocol for the original agent.

References

[1] S. Afantenos, N. Asher, F. Benamara, A. Cadilhac, C. Dégremont, P. Denis, M. Guhe, S. Keizer, A. Lascarides, O. Lemon, P. Muller, S. Paul, V. Popescu, V. Rieser, and L. Vieu, "Modelling strategic conversation: model, annotation design and corpus," in *Proceedings of the 16th Workshop on the Semantics and Pragmatics of Dialogue (Seinedial)*, Paris, 2012.

[2] D. Ariely, *Predictably Irrational: The Hidden Forces That Shape Our Decisions*. HarperCollins, 2008.

[3] A. Cadilhac, N. Asher, A. Lascarides, and F. Benamara, "Preference change," 2013, submitted.

[4] R. Coulom, "Efficient selectivity and backup operators in monte-carlo tree search," in *Computers and Games: Volume 4630 of Lecture Notes in Computer Science (LNCS)*, 2007, pp. 72–83.

[5] M. Guhe and A. Lascarides, "Trading in a multiplayer board game: Towards an analysis of non-cooperative dialogue," in *Proceedings of the 34th Annual Conference of the Cognitive Science Society*. Austin, TX: Cognitive Science Society, 2012, pp. 1626–1631.

[6] ——, "The effectiveness of persuasion in The Settlers of Catan," in *Proceedings of IEEE Conference on Computational Intelligence and Games – CIG 2014*, 2014.

[7] M. Guhe, A. Lascarides, K. O'Connor, and V. Rieser, "Effects of belief and memory on strategic negotiation," in *Proceedings of SEMDIAL 2013 – The 17th Workshop on the Semantics and Pragmatics of Dialogue, Amsterdam, 16–18 December 2013*, 2013, pp. 82–91.

[8] C. J. Hanna, R. J. Hickey, D. K. Charles, and M. M. Black, "Modular reinforcement learning architectures for artificially intelligent agents in complex game environments," in *Computational Intelligence and Games (CIG), 2010 IEEE Symposium on*, 2010, pp. 380–387.

[9] D. Kahneman and A. Tversky, "Prospect theory: An analysis of decision under risk," *Econometrica*, vol. 47, no. 2, pp. 263–291, 1979.

[10] M. Pfeiffer, "Machine learning applications in computer games," Master's thesis, Technical University of Graz, 2003.

[11] ——, "Reinforcement learning of strategies for Settlers of Catan," in *Proceedings of the International Conference on Computer Games: Artificial Intelligence, Design and Education, Reading, UK*, 2004.

[12] G. Roelofs, "Monte carlo tree search in a modern board game framework," 2012, research paper available at `umimaas.nl`.

[13] L. Savage, *The Foundations of Statistics*. John Wiley, 1954.

[14] Y. Shoham and K. Leyton-Brown, *Multiagent Systems: Algorithmic, Game-Theoretic and Logical Foundations*. Cambridge University Press, 2009.

[15] R. Sutton and A. Barto, *Reinforcement learning: An introduction*. MIT Press, 1998.

[16] I. Szita, G. Chaslot, and P. Spronck, "Monte-carlo tree search in Settlers of Catan," in *Advances in Computer Games*, H. van den Herik and P. Spronck, Eds. Springer, 2010, pp. 21–32.

[17] K. Teuber, *Die Siedler von Catan: Regelheft.* Stuttgart, Germany: Kosmos Verlag, 1995.

[18] R. S. Thomas, "Real-time decision making for adversarial environments using a plan-based heuristic," Ph.D. dissertation, Northwestern University, 2003.